4th IEEE COMSOC Intl. Workshop on Multimedia Communications, Monterey, CA, April 2-4, 1992

Protocols for Bandwidth-Constrained Multimedia Traffic

T.D.C. Little
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, MA, 02215 USA
tdcl@buenga.bu.edu

Abstract - To support time-dependent data, multimedia applications require network and operating system components that can ensure timely delivery of data. In addition, data of many applications such as videoconferencing and computer-supported collaborative work require coordination of multiple independent data sources. For single-medium connections, bandwidth utilization spans a moderate range. On the other hand, for multimedia connections, the large variation in data sizes of different data types indicates a large variation in utilized bandwidth during the life of a connection. Due to this variation, channel capacity can be periodically exceeded and some data may not be delivered in time to meet the intended play-out times. Furthermore, when independent sources and channels are considered, an intermedia synchronization mechanism must be provided that accommodates differing channel characteristics.

In this paper we describe a scheduling mechanism for supporting such bandwidth-constrained multimedia traffic and present protocols to provide intermedia synchronization of data originating from independent sources. The protocols provide functionality to establish and maintain individual connections with specified delay and bandwidth characteristics by the scheduling mechanism, and provide coordination of aggregate virtual connections.

1 Introduction

A unique characteristic of multimedia information systems is the requirement to support time-dependent data presentation (e.g., the playout of synchronized audio and video). Such a system must be able to overcome system delays caused by storage, communication, and computational latencies in the procurement of data for the user. These latencies are usually random in nature, being caused by shared access to common system resources. In a *distributed* multimedia information system, multiple data sources including databases, video cameras, and telephones can be connected to user workstations via high-speed packet-switched networks. System latencies in a this environment cause additional problems because data originating from independent sources can experience different delays require while still requiring intermedia synchronization upon playout at a single destination.

To support the presentation of time-dependent data, scheduling disciplines associated with real-time operating systems are required. Because multimedia data can tolerate some lateness in delivery without catastrophe, a statistical approach to the real-time scheduling of their retrieval and presentation can be used. For these data, real-time deadlines consist of the playout times for individual data elements that can be scheduled using a probability of lateness. Components of the system introducing random delays occur on the end-to-end path from data source to destination and include storage devices, the CPU, and the communications channel.

Support for time-dependent is a requirement for live data communications, data storage systems, and general distributed systems (see [1] for a survey of recent work). The work in data communications has usually been applied to live, single-connection, and periodic data sources such as packet audio and video [2-6] and for applications that do not exceed the capacity of the communication channel. In this work, reduction in delay variations at the receiver is achieved through the introduction of a time offset and buffering.

Work in supporting time-dependent data storage management concentrates on physical and conceptual models to enable data retrieval [7-14]. Conceptual models [11-14] support the orchestration of multimedia presentations, while physical models [7-10] are intended to achieve a balance between superior disk bandwidth and delay characteristics and effective disk space utilization for time-dependent data. Stored data differs from live data in that the times at which data are retrieved can be manipulated. Since data are not generated in real-time, they can be retrieved in bulk, well ahead of their playout deadlines. In contrast, live sources (e.g., a video camera), generate data at the same rate as required for consumption.

The work presented in this paper is intended for both stored and live-data applications. For live data sources, channel capacity cannot be exceeded, on average, for the life of a connection, or else the application is not viable. On the other hand, for stored-data, it is reasonable to specify the playout of many multimedia data elements in a concurrent fashion and therefore exceed available channel capacity. Since data can be retrieved in any order and at any time, stored-data facilitate smoothing of peak bandwidth utilization over time. The protocols presented here facilitate communication of both live-data traffic in a manner analogous to earlier approaches, and stored-data traffic by using a scheduling scheme described in Section 3.

We envision many application scenarios that are bandwidth limited. For example, an application can request a multimedia connection through an access mechanism to a shared network [15]. The request can be based on cost per bandwidth and will depend on available services. Another example is the delivery of multimedia services to the home via the local access loop at 1.5 Mbits/sec. [16]; a rate at which it is possible to deliver compressed video. However, when other traffic (e.g., images) is also simultaneously delivered, the bandwidth can be exceeded, resulting in the disturbance of video playout. Our scheduling mechanism is developed to prevent this disturbance.

The remainder of this paper is organized as follows. In Section 2, we discuss the properties of time-dependent multimedia data. In Section 3, we describe our approach to scheduling time-dependent traffic under a bandwidth constraint and practical considerations for applying the derived schedules. In Section 4 we present the protocols for intermedia synchronization. Section 5 concludes the paper.

2 Characteristics of Time-Dependent Multimedia Data

Time dependencies among multimedia data elements are often described using temporal-interval-based (TIB) modeling [11-13]. This approach associates a time interval to the playout duration of each time-dependent data object. For example, the playout of a sequence of video frames can be represented by using temporal intervals as shown in Fig. 1. An n-ary relationship called a *temporal relationship* (*TR*) can be identified among a set of intervals [11]; indicating whether they overlap, abut, etc. [17]. With a TIB modeling scheme, complex timeline representations of multimedia object presentation can be represented by relating a set of temporal intervals. Each interval can also be represented by its *start time* and *end time* via instant-based modeling. Therefore, for a sequence of data objects, we can describe their time dependencies as either a sequence of related intervals or as a sequence of start and end times. In the latter, the start times can be interpreted as *deadlines* as required for real-time scheduling. Fig. 2 shows the symbols that we associate with the intervals and deadlines in a time-dependent data specification. For each interval i we indicate its start time π_i and duration τ_i. The relative positioning of multiple intervals represents their time dependencies τ_δ^i. Their overall duration is defined by τ_{TR}.

Using the n-ary temporal relations, a conceptual storage schema is easily created (see Fig. 3). Furthermore, the deadlines necessary for real-time scheduling during playout can be derived from such a model, forming a monotonically increasing sequence [11].

In the presence of real system latencies, several additional delay parameters are introduced as shown in Fig. 4. In order to properly synchronize some data element x with a playout time π, sufficient time must be allowed to overcome the latency λ caused by data generation, packet assembly, transmission, etc., otherwise, the deadline π will be missed. If we choose a time called the *control time* T such that $T \geq \lambda$, then scheduling the retrieval at T time units prior to π guarantees successful playout timing. The *retrieval time*, or packet production time ϕ is defined as $\phi = \pi - T$. The case of synchronizing one event is equivalent to the problem of meeting a single real-time deadline, a subset of the real-time scheduling problem. For streams of data, the multiple playout times and latencies are represented by the sets $\Pi = \{\pi_i\}$ and $\Phi = \{\phi_i\}$.

Most single-medium synchronization schemes reduce delay variation in stream traffic by introducing a control time for all elements of a *stream* [2-6]. Elements of stream are generated at a source and experience a random delay before reaching their destination. They then require synchronization based on a playout schedule with deadlines π_i and playout durations τ_i. A suitable control time can be found for a stream of data and target percentage of missed deadlines based on such a delay characteristic [2-4]. Variation in both λ_i, the latencies of individual stream elements, and τ_i, their playout durations, can result in short term average or instantaneous need for buffering which is accommodated by T. For live sources, if ϕ_i are the packet production times, then $\phi_i = \pi_i - T$, $\forall i$, where π_i are the playout times. The playout sequence is merely shifted in time by T from the generation times. Furthermore, playout durations are typically uniform ($\tau_i = \tau_j$, $\forall i, j$), and delay variation governs the buffering requirement. In the general case, we are presented with arbitrary streams characterized by π_i, λ_i, and τ_i and would like to determine the amount of buffering necessary to maintain playout timing with a given probability of failure.

When a stored-data source is considered, it is reasonable for the playout schedule to indicate short-term overload of the channel capacity. In this case, previous scheduling policies are deficient because they assume that the generation of packets is at a rate equal to the consumption rate. In the next section we describe a framework for dealing with this kind of data source.

3 Scheduling with a Bandwidth Constraint

Because we assume that the channel capacity can be exceeded, a method is sought for smoothing the overloaded capacity during the life of a connection that maintains playout timing. In this section we summarize our scheduling approach [1]. We use *a priori* knowledge of data traffic characteristics to facilitate scheduling, when available. Otherwise, the statistical nature of the multimedia traffic source is anticipated through existing statistical methods. Essentially, the dynamic bandwidth requirements of a multimedia object are fit into finite resources of delay and channel capacity rather than the resources dictating the feasibility of the application. The result is that delays are traded-off for the satisfaction of a playout schedule, even when the capacity of the channel is exceeded by the application. The essence of the scheduling scheme is described next.

3.1 Synopsis of the Scheduling Approach for Single Connections

Depending on their size and playout times, transmission of a sequence of objects is either back-to-back or introduces slack time. We define an optimal schedule for a set to be one that minimizes the object's control times. Two constraints determine the minimum control time for a set of objects. These are the minimum end-to-end delay (MD constraint) per object, and the finite capacity (FC constraint) of the channel. The MD constraint simply states that an object cannot be played-out before arrival. This constraint must always be met. The FC constraint describes the relation between a retrieval time and its successor when the channel is busy and accounts for the interarrival time due to transit time and variable delays. It also represents the minimum retrieval time between successive objects.

Given the characteristics of the channel and of a composite multimedia object (D_v, D_p, D_t, C, S_m, π_i, $|x_i|$, $P(fail)$, corresponding to variable, fixed, and channel transmission delays, channel capacity, packet size, playout deadline, object size, and probability of lateness), a schedule is constructed [1]. Construction begins by establishing an optimal retrieval time for the final, *m*th object, i.e., $\phi_m = \pi_m - T_m$. The remainder of the schedule can be determined by iterative application of the MD and FC constraints for adjacent objects. The resultant schedule indicates the times to put objects onto the channel between the source and destination, and can be used to establish the worst case buffering requirement.

3.2 Implementation Tradeoffs

In this section we show how the derived schedule Φ is used. The sender can use it to determine the times to put objects onto the channel. The receiver can then buffer the incoming objects until their deadlines occur. In this case the source must know Φ, the destination must know Π, and the overall control time is indicated by T_1, the computed delay of the first element. This scheme is supported by appending π_i onto objects as they are transmitted in the form of time stamps. Another way to use Φ is to determine the worst-case skew between any two objects as $T_w = \max(\{\pi_i - \phi_i\})$, and then to provide T_w amount of delay for every object. Transmission scheduling can then rely on a fixed skew T_w and only Π, i.e., all items are transmitted such that π_i is in the interval $(t \leq t + T_w)$.

The first method minimizes both buffer utilization and delay since the schedule is derived based on these criteria. Furthermore, required buffer allocation need not be constant but can follow the buffer use profile. The second method provides unnecessary buffering for objects, well ahead of their deadlines, and requires constant buffer allocation. However, it provides a simpler mechanism for implementation. Consider a long sequence of objects of identical size and with regular (periodic) playout intervals. For this sequence, $T_1 = T_w$, and $\phi_i = \pi_i - T_w$, assuming the channel capacity C is not exceeded. In this case, the minimum buffering is also provided by the worst-case delay. Such a sequence describes constant bit rate (CBR) video or audio streams which are periodic, producing fixed size frames at regular intervals. Rather than manage many computed deadlines/sec. (e.g., 30/sec. for video), a transmission mechanism more simply sequentially transmit objects based on sequence number and T_w.

When data originate from live sources, the destination has no control over packet generation times, and sufficient channel capacity must be present to preserve the real-time characteristic of the streams. In this case, the control time can be determined from the size, delay, and channel capacity requirements of a representative data object, (e.g., a CBR video frame), and only provides jitter reduction at the receiver. For variable bit rate (VBR) objects, the source data stream is statistical in size and frequency of generated objects, and we apply a pessimistic worst-case characterization of the largest and most frequent VBR object to determine the control time. $\{\phi_i\}$ defines the packet production times, and playout times are $\pi_i = \phi_i + T$, as done in previous work. This service can be supported by appending timestamps, in the form of individual deadlines, π_i, to the transmitted objects.

In the general case, a playout schedule is aperiodic, consisting of some periodic and aperiodic deadlines. Typically, during class decomposition, these are isolated, and one of the two scheduling schemes above can be applied. If both exist in the same playout schedule Π (e.g., if classes are not decomposed), then the derived schedule Φ can be used by choosing a new control time T_E such that ($T_1 \leq T_E \leq T_w$), and by dropping all deadlines ϕ_i from Φ such that $T_E \geq \pi_i - \phi_i$. The result is a culled schedule Φ reflecting the deadlines that will not be satisfied by simply buffering based on T_E. By choosing T_E to encompass a periodic data type (e.g., video), the burden of managing periodic deadlines is eliminated, yet aperiodic objects, requiring extensive channel utilization, can be dealt with using $\phi_i - T_E$, where $\phi_i - T_E > 0$.

4 Protocols for Multimedia Synchronization

Two levels of synchronization and transmission service are identified to support general purpose multimedia communications, based on the scheduling framework outlined in Section 3. These are for *independent*-connection traffic (IC) and *aggregate*-connection traffic (AC) consisting of multiple synchronized IC traffic. For IC traffic, a service mechanism is defined for the support of applications requiring synchronization of sequences of data of the same type or origin. This mechanism provides synchronized output for a set of data and their playout schedules. For AC traffic, temporal information in the form of a TIB specification scheme is utilized to produce playout schedules for synchronized communication of multiple classes of data. By decomposing these two levels of functionality, the IC protocol mechanism is isolated from inter-class dependencies and the TIB specification, and the AC protocol mechanism achieves a abstract interpretation of the objects and their origins rather than their transport mechanism.

Because the IC and AC functionality is separated, the AC can provide a more application-oriented interface including operations of complex object retrieval from multiple sources across a network for playout at a single site. The AC interface to the application takes as input a selected object representing the aggregation of complex multimedia presentation requiring synchronization, and returns streams of distinct synchronized data traffic which can then be routed to specific workstation output devices for presentation.

4.1 The IC Protocol

The IC protocol requires as input a set of multimedia objects O_i, their playout times π_i, their aggregate start time, and the desired probability of data lateness. The protocol can then provide a predicted end-to-end control time T and data transport mechanism for the set of data elements, using the scheduling mechanism of Section 3. The protocol performs the following operations:

(1) Reservation and negotiation for channel capacity C.
(2) Identification of current channel delay parameters.
(3) Computation of retrieval schedule and required buffers based on (2) [1].
(4) Data initiation and transfer.

(1) and (2) are dependent on the access mechanism (e.g., [15]). (3) is described in [1]. The data initiation operation (4), allows a synchronous start of multiple aggregate connections. Within the IC protocol, synchronous data retrieval and transfer is provided by the following algorithm which sends data from source to destination:

```
while $\exists i$ ($sent_i$ = false) do
  if $clock \geq \phi_i + start$ and $sent_i = false$ then
    {send object $i$ to destination with appended playout deadline $\pi_i$}
    $sent_i := true$
  end
  if ($clock \leq \pi_i + start \leq clock + T_E + start$) and $sent_i = false$ then
    {send object $i$ to the destination with appended playout deadline $\pi_i$}
    $sent_i := true$
  end
end
```

The algorithm operates by iterating on the set of objects waiting for transmission, assuming initially that no objects are sent ($\forall i$, $sent_i = false$). The first conditional statement in the algorithm tests for impending retrieval deadlines for objects in the culled retrieval schedule (see Section 3.2). The second conditional statement identifies the retrieval time of the remaining objects not in the retrieval schedule.

At the receiving site, data are buffered and released to the application at the times indicated by the appended deadlines π_i. In the next section, we describe the services provided by the AC protocol and show how they interact with the IC protocol.

4.2 The AC Protocol

The AC protocol takes a selected object, decomposes it into elements of distinct traffic classes for the IC protocol, and then provides inter-class synchronization. A typical application can use the AC service for stored, pre-orchestrated presentations or for live sources such as videophones. For these applications, a TIB specification can be maintained in a database describing the object whether live or stored. Once an object is identified, the remaining steps required to retrieve and present the sought multimedia object are provided by the AC. These include:

(1) Traversal of an object's temporal hierarchy [11].
(2) Decomposition of traffic classes based on type and location.
(3) Generation of playout schedules from temporal relations [11].
(4) Invocation of IC protocol to determine individual control times (see Section 4.1).
(5) Identification of maximum control time for intermedia synchronization.
(6) Initiation of synchronous data transfer.

Steps (1) - (4) are described elsewhere. Step (4) returns multiple control times, T_E, in response to multiple calls to the IC protocol. It also indicates whether the required resources are available. The AC protocol then determines an overall control time (step 5) for the synchronous initiation of data transfer at each of the multiple IC invocations. The overall control time is found as $\max(\{T_{E_j}, \forall j\}$, where j ranges across the set of IC invocations.

Once the maximum control time is determined, the IC invocations can be notified, via step (6), as to when to initiate their data transfer schedules. These are determined from the overall control time T_o by adding the current time and subtracting the local control time:

$$start_j = clock + T_o - T_{E_j}$$

After initiation, multiple invocations of the IC protocol transfer data from sources to the destination with attached deadlines for playout. At the receiver (destination), the arrived data are released to the application process per the individual schedules, to be routed to the appropriate output devices.

For applications requiring intermedia synchronization of live sources, playout deadlines are generated on-the-fly from the stream of packets emanating from the source. Data of this type are handled by characterizing their statistical size and frequency and choosing an appropriate control time [2-4]. However, these live applications cannot benefit from time-shifting or buffering of any kind if the average required channel capacity exceeds the available channel capacity. With respect to the IC protocol, only random channel delay variations are resolved. We now illustrate an example of the AC decomposition and the IC scheduling.

5 Example

Fig. 5 illustrates the time-dependencies a complex multimedia orchestration from a stored-data source, represented by an OCPN [12]. The shaded portion of the figure represents a sequence of full-motion video images from some source A. The video is assumed to be compressed and has a nominal playout rate of 30 frames/sec. ($|x_i| = 1 \times 10^6$ bits and $\tau_i = 1/30$ sec.). The remaining portion of the figure shows a more complex group of objects from some source B which exceeds channel capacity. In this case, audio (64 Kbits/sec.), video and images (1024 x 1024 pixels x 8 bits/pixel) are coordinated in a multimedia presentation originating from source B, but unlike source A, video and audio data elements are retrieved in 5 and 10 sec. segments, respectively. The following conditions are assumed for both communications channels: $C = 1.5$ Mbit/sec., $S_m = 8192$ bits, $D_v = 50$ μsec., and $D_p = 100$ μsec. After the two traffic classes are isolated by the AC mechanism, the results of the IC scheduling algorithm are, for source A:

$$\Pi = \{0.0, 0.033, 0.067, 0.10, 0.13, 0.17\} \text{ sec.}$$
$$\Phi = \{-0.028, 0.0057, 0.039, 0.072, 0.11, 0.14\} \text{ sec.}$$
$$K = \{0, 0, 0, 0, 0, 0\} \text{ bits}$$

and for source B:

$$\Pi = \{0, 0, 0, 5, 10, 10, 15, 20, 20, 20, 25, 30, 30, 35\} \text{ sec.}$$
$$\Phi = \{-9.5, -6.1, -5.7, 0.0, 0.0, 3.4, 3.8, 7.2, 10.5, 11.0, 16.7, 21.6, 26.2, 26.6\} \text{ sec.}$$
$$K = \{0, 50, 100, 184, 50, 100, 184, 184, 100, 184, 184, 50, 50, 100\} \times 10^5 \text{ bits}$$

These results indicate that there is ample time to transmit the specified video traffic from source *A* without buffering (*K* buffers) beyond the object in transit, i.e., the size and playout timing is not affected by the channel capacity constraint. For source *B* the scheduling algorithm produces a retrieval schedule based on scheduling adjacent objects in the channel. The results indicate that an initial delay of 9.5 sec. is required prior to beginning playout at the receiver. Finally, in order for the AC protocol to provide intermedia synchronization between these independent sources, an overall control time of $T_0 = \max\{0.028, 9.5\}$ sec. is required, which can be used for generation of the value of *start*.

6 Conclusion

A significant requirement of broadband multimedia communications services is the ability to synchronize multiple streams of heterogeneous data necessary for applications including multimedia databases, videoconferencing, and computer-supported collaborative work. In this paper we have described communications protocols for both live and stored time-dependent traffic types under limited bandwidth conditions. The protocols provide functionality to establish individual connections with specified delay and bandwidth characteristics, and provide coordination of aggregate virtual connections from independent traffic sources. For stored-data sources, the protocols assume *a priori* knowledge of the characteristics of multimedia data elements to facilitate efficient use of channel capacity. However, they are also amenable to live-data sources. The approach utilizes a timing specification in the form of a set of monotonically increasing playout times which are derivable from a temporal-interval-based timing specification. From the timing specification a retrieval schedule is computed based on the characteristics of the limited-capacity resource. The examples presented illustrate the utility in the mechanism for applications which have channel capacity constraints.

7 References

[1] Little, T.D.C., Ghafoor, A., "Scheduling of Bandwidth-Constrained Multimedia Traffic," to be published in *Computer Communications* (Special Issue: Multimedia Communications), May 1992.

[2] Barberis, G., "Buffer Sizing of a Packet-Voice Receiver," *IEEE Trans. on Comm.*, Vol. COM-29, No. 2, February 1981, pp. 152-156.

[3] Montgomery, W.A., "Techniques for Packet Voice Synchronization," *IEEE J. on Selected Areas in Comm.*, Vol. SAC-1, No. 6, December 1983, pp. 1022-1028.

[4] De Prycker, M., Ryckebusch, M., Barri, P., "Terminal Synchronization in Asynchronous Networks," *Proc. IEEE Intl. Conf. on Comm.*, Seattle, WA, June 1987, pp. 800-807.

[5] Naylor, W.E., Kleinrock, L., "Stream Traffic Communication in Packet Switched Networks: Destination Buffering Considerations," *IEEE Trans. on Comm.*, Vol. COM-30, No. 12, December 1982, pp. 2527-2524.

[6] Ades, S., Want, R., Calnan, R., "Protocols for Real Time Voice Communication on a Packet Local Network," *Proc. IEEE INFOCOM '87*, San Francisco, CA, March, 1987, pp. 525-530.

[7] Rangan, P.V., *et al.*, " A Testbed for Managing Digital Video and Audio Storage," *Proc. 13th Symp. on Operating Systems Principles (SOSP'91), Operating Systems Review*, Vol. 25, No. 5, October 1991, pp. 81-94.

[8] Gemmell, J., Christodoulakis, S., "Principles of Delay Sensitive Multi-Media Data Retrieval," *Proc. 1st Intl. Conf. on Multimedia Information Systems '91*, Singapore, January 1991, McGraw-Hill, Singapore, pp. 147-158.

[9] Yu, C., Sun, W., Bitton, D., Yang, Q., Bruno, R., Tullis, J., "Efficient Placement of Audio Data on optical Disks for Real-Time Applications," *Comm. of the ACM*, Vol. 32, No. 7, July 1989, pp. 862-871.

[10] Wells, J., Yang, Q., Yu, C., "Placement of Audio Data on Optical Disks," *Proc. 1st Intl. Conf. on Multimedia Information Systems '91*, Singapore, January 1991, McGraw-Hill, Singapore, pp. 123-134.

[11] Little, T.D.C., Ghafoor, A., Chen, C.Y.R., "Conceptual Data Models for Time-Dependent Multimedia Data," *Proc. 1992 Workshop on Multimedia Information Systems*, Tempe, AZ, February 1992, pp. 86-110.

[12] Little, T.D.C., Ghafoor, A., "Synchronization and Storage Models for Objects," *IEEE J. on Selected Areas in Comm.*, Vol. 8, No. 3, April 1990, pp. 413-427.

[13] Herrtwich, R.G., "Time Capsules: An Abstraction for Access to Continuous-Media Data," *Proc. 11th Real-Time Systems Symp.*, Lake Buena Vista, FL, December 1990, pp. 11-20.

[14] Stotts, P.D., Furuta, R., "Petri-Net-Based Hypertext: Document Structure with Browsing Semantics," *ACM Trans. on Office Automation Systems*, Vol. 7, No. 1, January 1989, pp. 3-29.

[15] Lazar, A.A., Temple, A., Gidron, R., "An Architecture for Integrated Networks that Guarantees Quality of Service," *Intl. J. of Digital and Analog Cabled Systems*, Vol. 3, No. 2, 1990.

[16] Rosenberg, J., Cruz, G., Judd, T., "Presenting Multimedia Documents Over a Digital Network," *Proc. 2nd Intl. Workshop on Network and Operating Support for Digital Audio and Video*, Heidelberg, Germany, November 1991.

[17] Allen, J.F., "Maintaining Knowledge about Temporal Intervals," *Comm. of the ACM*, November 1983, Vol. 26, No. 11, pp. 832-843.

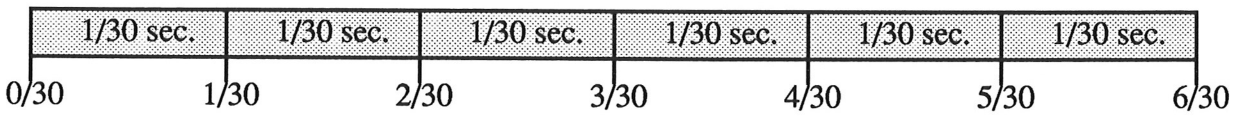

Fig . 1. Temporal-Interval-Based Specification for Video Frames

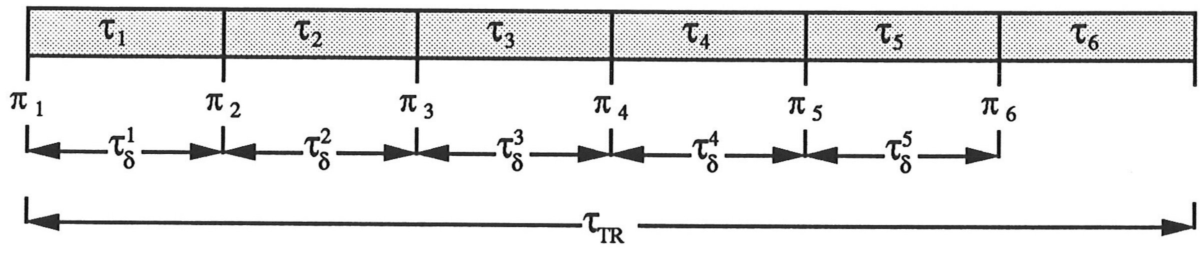

Fig. 2. Timing Parameters

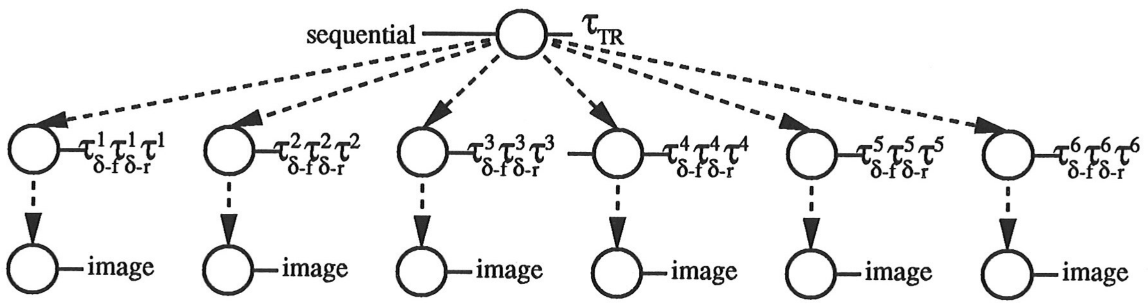
sequential
τ_{TR}
$\tau_{\delta\text{-}f}^1 \tau_{\delta\text{-}r}^1 \tau^1$
$\tau_{\delta\text{-}f}^2 \tau_{\delta\text{-}r}^2 \tau^2$
$\tau_{\delta\text{-}f}^3 \tau_{\delta\text{-}r}^3 \tau^3$
$\tau_{\delta\text{-}f}^4 \tau_{\delta\text{-}r}^4 \tau^4$
$\tau_{\delta\text{-}f}^5 \tau_{\delta\text{-}r}^5 \tau^5$
$\tau_{\delta\text{-}f}^6 \tau_{\delta\text{-}r}^6 \tau^6$
image
image
image
image
image
image

Fig. 3. Conceptual Schema for Video Frames

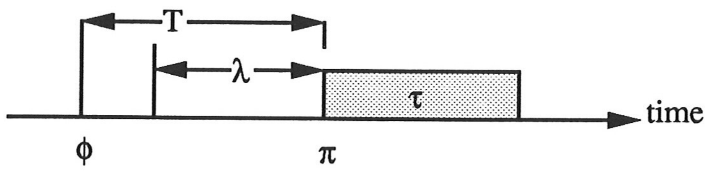

Fig. 4. Timing Parameters Including Latency

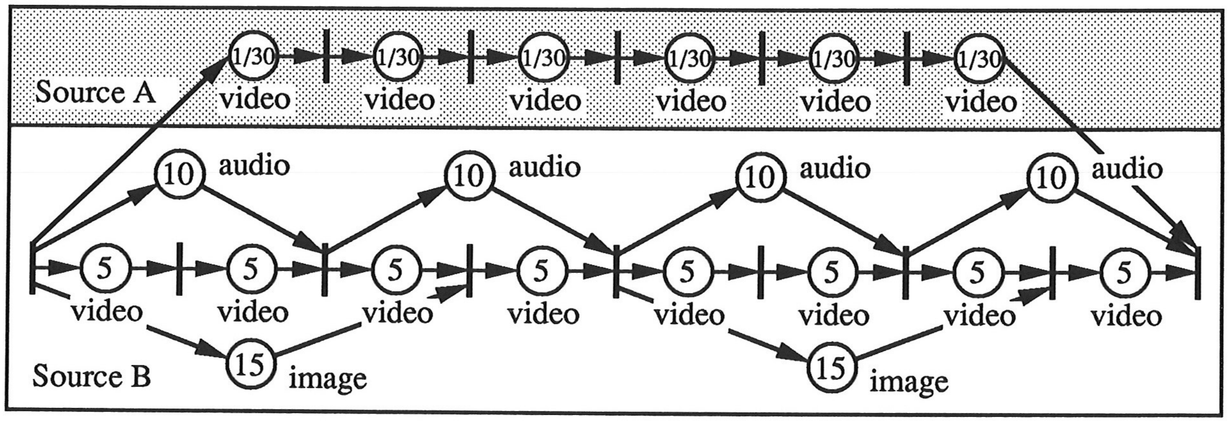

Fig. 5. Audio, Video, and Image OCPN Timing Specification